Domisilica: Providing Ubiquitous Access to the Home

Jen Mankoff and Gregory D. Abowd
Graphics, Visualization and Usability Center
College of Computing
Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332-0280
{abowd, jmankoff}@cc.gatech.edu

ABSTRACT
The Domisilica project is exploring future computing environments centered around the home. We envision a future in which objects in the home, such as appliances and rooms, are enhanced with computational capabilities that make them accessible away from the home. Our approach has been to add computational power to real world objects located in the home, and to provide a centralized model of the home which can be used to integrate a wide variety of services and is accessible remotely. This paper focuses on issues of interface modes and accessibility which arose as we developed Domisilica. These include extending the affordances of real world objects to provide new services, providing remote access through a variety of connections both impoverished and broad, and providing universal access for all types of people including people with disabilities, children, and older adults.

KEYWORDS: Computing in the home, ubiquitous computing, dynamically generated interfaces, multimodal interfaces

INTRODUCTION
Think of your home as an interface to the information and objects associated with it. Now imagine having access to that interface from any computer. Domisilica is a project which not only provides a way for you to access the information which is part of your home, but also allows you to associate additional virtual information of all sorts with your abode. Virtual information might include the purchase date of groceries, shopping lists, or web pages.

We have chosen the kitchen as the place to begin our exploration of computing in the home. The refrigerator, for example, has many uses beyond keeping food cool in the common household: making shopping lists, leaving notes, and posting information and pictures. Domisilica's computational capabilities and network connection allow us to expand these tasks. By attaching a computer display to the front of the refrigerator, we can display virtual notes and web pages which have been "posted" onto the refrigerator. A remote user can view the contents of the refrigerator in a GUI or hear them over the phone. That user can create notes and post them to the refrigerator, or run an application which makes use of the kitchen inventory system to generate a shopping list or produce a potential menu for dinner. See Figure 1 for a idea of the kind of remote interaction we currently support in Domisilica. [1]

Classes of activity supported by Domisilica
The types of activities which can benefit from a computing system in the home include communication (such as real-time chatting, and asynchronous notes), inventory management (such as keeping track of the contents of the refrigerator), and control (such as answering the door or turning on and off appliances). Systems like CEBus [1] and X10 [2] can be used for control activities. Domisilica provides support for communication and inventory management as well.

Currently, computing in the home takes two forms: It is centralized in one box where occupants go to run various applications, or it is distributed in small, autonomous chunks in appliances such as microwaves, VCRs, and security/climate control systems. We envision a future in which devices currently common in households are augmented with additional information and functionality, and networked together to provide information and services to remote users.

This scenario presents several interesting problems:

1. Once a real world object or device has been augmented with new services and information, how do you make the user aware of these services, and how do you make these services available to a user through the object itself ("on location")? We believe that this is best done by extending existing affordances of the object when possible, and adding new ones when necessary. As an example, consider a front door which has been augmented with an outdoor camera. One way of extending the affordances of the door might be

[1] See the included video, and the Domisilica web pages (http://www.cc.gatech.edu/fce/domisilica) for a demonstration of this scenario.

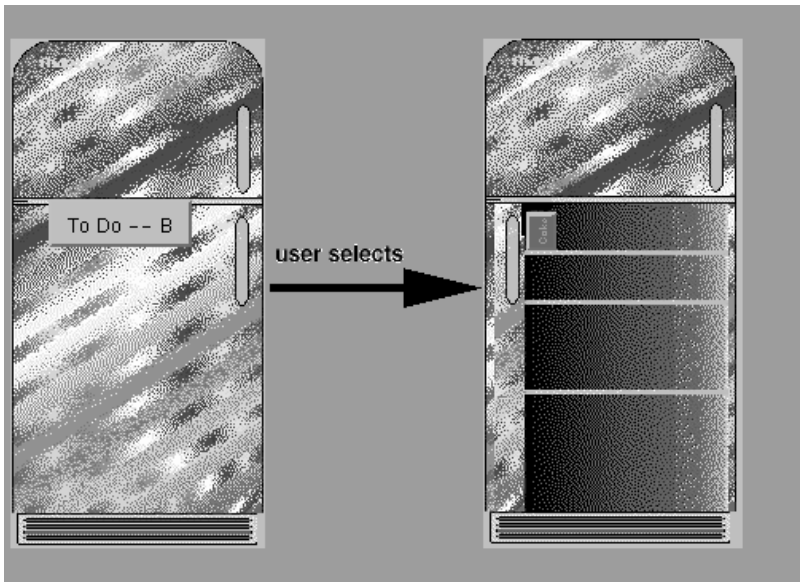

Figure 1: A snapshot of remote interaction with the visual interface to Domisilica's refrigerator. On the left is an image of the refrigerator with a virtual note posted on it. When the user selects the door with the mouse, the contents of the refrigerator (A lone can of soda) are revealed.

to replace the peephole with a "hole" that can be rotated and slid around the door in order to control the view of the camera.

2. Universal accessibility is very important both locally and remotely. Homes are lived in by people of all sorts and sizes, including people who are handicapped, older, very young, and people who may have very little computer experience, etc. Our system also needs to be accessible remotely to people without computers and people with very low bandwidth connections and limited services, as well as computing professionals with high-bandwidth personal machines. Here again, the accessibility of the system to different people is an important consideration.

Overview of Paper

In the next section we discuss related projects. Following that, we introduce the Domisilica system as it currently stands. The next section discusses the variety of interfaces which are needed for ubiquitous access to and in the home. In order to support the range of interfaces required to solve the problems described above, we are building a set of toolkits for integrating and creating different types of UIs. After discussing these toolkits, we conclude with a section on future work.

RELATED WORK

This work grew from the PALplates project [12] (done at FXPAL) which consists of a remotely accessible model of a real space (an office place) and objects in that office. PALplates only supported one mode of interaction (a GUI interface). Touch screen terminals were placed around the office in order to provide access to new services "on location". Another project which helped to guide our initial approach was the Jupiter project [14], which implements remote access from home to real world office spaces. Jupiter was built on the assumption that its users would have a high-quality connection (good enough to support audio and video multicasting), an assumption which cannot be made for users of a home, who may include people members with access to telnet or email at most.

While people have studied computing in the home [18, 11], the focus has generally been on technologies imported from the office place (such as email, the Web, and computer-centric tasks like word-processing). One exception to this is the Neural Network House [13], a home which manages climate control and learns about its user's habits over time. While this research explores a climate control service in depth, it does not address the interface issues we are contending with.

Both the ubiquitous computing community and many entrepreneuring technophiles have explored ways of adding new computing power throughout our physical environments. (See the July 1993 CACM (titled "Back to the Real World").) Augmented reality [22, 6, 19, 21], represents one approach to giving users access to new services and information. The DigitalDesk project [21] is a good example of how the affordances of paper can be extended to provide new services. The ParcTab/Pad ubiquitous computing project [20] involves adding new computing objects to the environment rather than extending already existing objects into the computing world. Fitzmaurice [6] describes a portable, personal palmtop display which can be used to view augmented aspects of real world objects, and Feiner et. al [5] use a wearable display

to show virtual information overlayed on real objects. While these are a valid approaches, they require the user to carry the interface. As an alternative, we have chosen to build the input and output into the augmented device. This type of interface is exemplified by the work of Ishii et. al in graspable user interfaces and tangible bits [9, 7].

Much of the research done in supporting disabled users [10, 17, 4, 16] is very applicable to Domisilica. Work involving blind users can support both our goal of universal access, and can also be applied to the development of non-visual interfaces such as phone-based access. The Dual project [17] provides a way of generating both a visual and non-visual interface to the same application in an integrated fashion, while Mercator [4] automatically generates an audio interface from an X-based GUI. Emacspeak [16], similar to Dual, can present both visual (text-based) and non-visual interfaces simultaneously. Domisilica follows the philosophy of Dual: that non GUI interfaces are best developed independently with metaphors appropriate to the medium. We feel, however, that there are many different modes of UI's which should be supported, each with different metaphors, not just two. These include GUIs; visual but non-graphical UIs (eg MOOs[2], emacs); audio and phone-based UIs (eg conversational metaphors), and many others.

THE DOMISILICA SYSTEM

Domisilica consists of a central database which stores a model of the home; a Java toolkit for developing UI's; several devices for use in the kitchen; a system for converting text-speech for audio interfaces; and a portable device which can be used to hear audio. The system is installed and can be accessed remotely by anyone on our internal web server. The real world side of the system is currently set up only for demos, but we are building a more permanent installation for it in a Home Infrastructure laboratory which includes a living room, kitchen, and bedroom. Figure 2 provides an overview of the system.

The database is a MOO [3], running on the LambdaMOO Server. It contains a model of the home, but also includes virtual data which has no real world correspondence (this is what we need to build interfaces to in the home), even though it may be associated with specific appliances, etc. An example of virtual data might be an electronic note which has been posted on the "front" of the virtual refrigerator. In addition, much of the state of the real world is not modeled in the database. Devices and users can connect to the database to run functions, update status, and access state or other data. In order to connect to the server, a device, person, or user interface must be able to open a socket connection. The server can also open outbound socket connections to other programs when necessary.

The Java toolkit provides support for building interactors for objects in the database (which generally correspond to devices, places, or services in the home). It also supports a communication protocol for requesting information and receiving events from the MOO. The GUI aspects of the package are built on top of SubArctic [8]. (SubArctic, like Mercator [4] can generate audio interfaces, but the preferred method for non-visual access is via the conversational, text-only interface automatically provided by the MOO). The GUI includes a window (the "Application Manager") which will dynamically load all of the interactors associated with objects in the MOO model which are visible to the user. This means that users can add new functionality to the database simply by writing a java class (an interactor, in the GUI case) which implements that functionality, and setting a special property of the corresponding database object to contain the name of the new class. The standard GUI system will then dynamically load that new interactor, incorporating it into the user's display on the fly. Our GUI application manager can be run from the WWW as an applet or locally as a Java application.

As an example of the real world side of things, we have implemented a kitchen inventory system which uses a bar code reader and an image recognizer to identify food items as they are unpacked. Note that while this may support additional functionalities, it is not in and of itself a way to access new data or call new functions. It is simply a way of informing the computer of changes in state. Ideally this should be done auto-magically. As the system stands, it still requires some overhead from the user, who must run each food item through the scanner or recognizer before putting it away. We feel that this is not unreasonable since most people start at a central place (a pile of bags) when unloading groceries, provided we can show them some gain for the extra work. Ultimately we hope to embed scanning technology in the places where food is stored. Supermarkets are already working on technology which can scan a whole shopping cart at once. Why not a whole pantry or refrigerator?

When designing this system, we chose a centralized model of the real world because it simplifies the task of providing a representation of the current state of services available to the user. There are arguments for additional robustness in distributed systems, but Domisilica can also be integrated with distributed computing in the house if each individual appliance is responsible for updating their status in the database with log data at regular intervals.

PROVIDING ACCESS TO COMPUTATIONALLY ENHANCED REAL WORLD OBJECTS

Once real world objects have been computationally enhanced (extended into the computing world), their user interfaces must also be enhanced. This task has two parts: on location access in which we follow the approach of Ishii et. al. [9]; and remote access, which draws on research done for the Jupiter project and on the PALplates project. Both types of

[2] a MOO is object-oriented, extensible database built to support "shared computing with a powerful real world metaphor." [3]

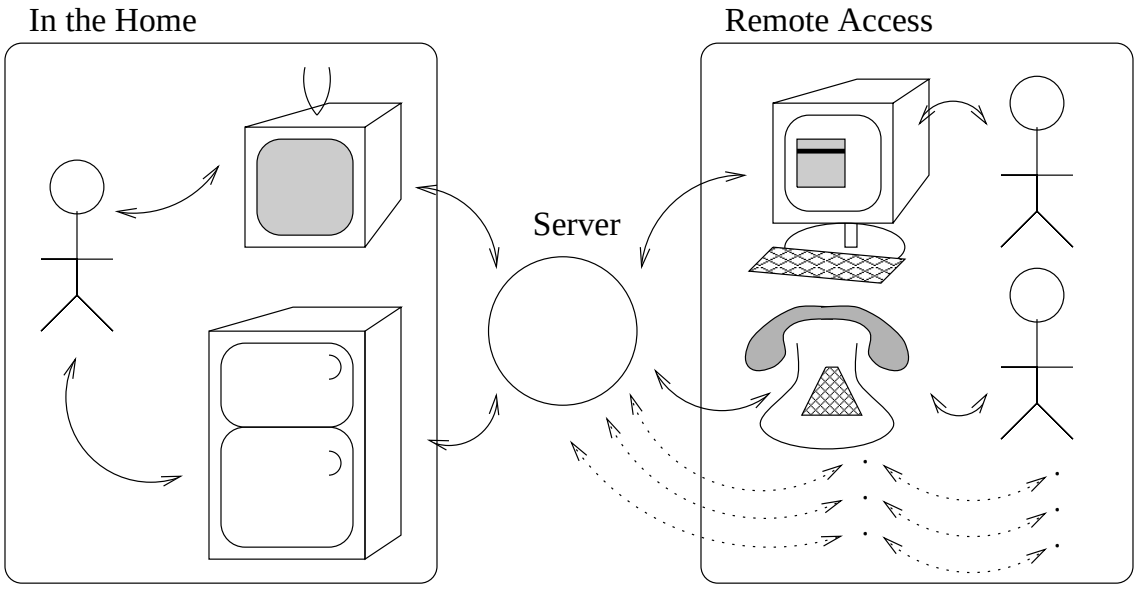

Figure 2: The connections between the server, the real world, people (on-location and remote), and devices

access must support users of all ages and abilities. A home must be both safe (secure) and accessible to any potential (and valid) users.

On Location Access

The basic goal in local access is to extend the affordances of real world objects [15] whenever possible to support their enhanced functionality. For example, in order to allow people to view and leave electronic notes on the door of their refrigerator, it makes sense to embed a touch screen or pen-operated display in that door. We are currently building a pen computer into a refrigerator door. Even better might be a set of portable notes which can be physically carried from location to location but can display electronic data.

Remote Access

In order to provide universally accessible remote access, we need to support the variety of modalities via which a user might wish or need to use when connect to our system. For example, a user with RSI (repetitive strain injury) might find a GUI output plus voice input system most accessible. A user without a computer will almost certainly have access to a telephone.

The following set of modalities are currently supported:

GUI for full-bandwidth connections

text for limited-bandwidth connections or small displays

audio output for low-vision users, to provide "atmosphere", and eventually for any situation where audio might be more appropriate.

audio-to-phone output for receiving status updates via a portable system (we are working on input, at which point users will be able to phone the system and interact with it using a combination of DTMF and voice).

Other modalities we plan to support include various combinations of audio output, pointer input, typing, voice-as-data, and recognized voice. Uses of voice-as-data include multimedia notes and direct conversation between remote and on-location users of the system. Many of these modalities will also be helpful in on-location interface development.

ONE SYSTEM TO SUPPORT MULTIPLE MODALITIES, METAPHORS, AND USERS

The implementation of the large variety of user interfaces described above sounds like a daunting task. However, it can be greatly simplified, if we begin to separate interface from functionality and to support development of multiple interfaces by encapsulating non-modality-specific interface features in common computational objects (see Dual [17]).

This approach is facilitated by the fact that all of our functionality is stored in a database (MOO objects are programmable, so in addition to storing data, they store *functionality*). Interfaces are just a view of the properties of objects (data) plus interactors for accessing their functionality. One very important aspect of this work is that different interfaces present the user with a metaphor tailored for the modality of interaction [17, 23, 10]. We have a slight advantage here over other applications; there is an obvious semantic metaphor that holds the system together: the mapping to the real world (rooms in a house containing various appliances). This metaphor is used by our version of a Window Manager or "Application Manager". It gives the user a way to access any of the many applications available in a given room and to navigate to different rooms. Even so, this metaphor must be tailored to the situation, and different representations of a

room or set of rooms are appropriate in different settings. For example, one interface may draw a three-dimensional picture of a room, while another might simply map real world places to folders and appliances to recognizable icons, and a third might present a conversational interface involving textual descriptions. Also, the real world mapping metaphor does not help as much at the application level. Each application may be built by a different person, and must support multiple modalities and metaphors. This is especially true for functions which have been added to the real world – for example, a function which visualizes the changes in population of a room over time might take a very different form in a GUI than "on location" or in an audio interface. Our solution to this problem is to provide toolkits for different modalities. Toolkits for GUI, audio, and text-based access are described in the following sections.

Toolkit for building GUI interfaces

We have implemented a standard set of interactors corresponding to the following MOO objects: "rooms", "maps", "containers", and "things" (objects that cannot contain other objects)[3]. Programmers can subclass from these interactors to provide special functionality, or use as-is if they simply wish to represent state information about the domicile. Each MOO object is represented on-screen either as a dynamically-loaded Java class, or a pointer to a URL. Most MOO objects are shown as icons which can be used to identify the real world object corresponding to that MOO interface object. The icon, Java classname, and other UI information, along with functions implementing any special functionality are all stored in the MOO. The Java toolkit simply parses that data (we provide a parser class and a connection to the MOO along with a protocol for requesting and receiving information) and makes the appropriate changes in the UI.

Text-based access

The MOO database server (LambdaMOO server) was initially built specifically for text-based access. The metaphor supported by LambdaMOO is that of a text-based virtual space with a conversational, semi-natural language interface. The user sees descriptions of: the room they are in, any objects or users also in that room, and any available exits. They can ask for more details, run commands, converse with other users, or navigate to other rooms. In our system, these rooms correspond to real world places which the user can visit in person (or may be standing in if they are doing on-location text-based access).

Audio access

The semi-natural language interface (which has a limited vocabulary) and text-based rooms metaphor are ideal candidates for audio input and output respectively. Although we have not yet constructed a system for audio input, we do have audio output working. Currently, audio can be played on a speaker or sent to a portable phone. This is useful in non-interactive situations for announcements and monitoring, and can be combined with other modes of input such as a keyboard for full interactivity. Audio output has been integrated with a system used by blind users called Emacspeak [16].

[3] The javadocs for this toolkit are available at http://www.cc.gatech.edu/fce/domisilica/docs/packages.html

FUTURE WORK AND CONCLUSIONS

We see potential for expansion of many different aspects of Domisilica in the future. A short list of some of these areas follows:

Persistence in UI's : The Domisilica system is a persistent database which may have users connected at any time, or include a persistent UI. When a new appliance is added, or an interface to an existing appliance is updated, currently connected users should not have to exit and reconnect, and persistent interfaces should automatically update to reflect the new state of the database.

Security : If Domisilica is to come into use in a real house, we need to make some guarantees about security, and our user interfaces need to support those guarantees. Currently, users are required to have a password to access the system, but that's the extent of our security. In the future, we need to make it possible for members of the same household to have secure spaces which other family members cannot enter, and to provide more stringent authentication in order to make the system more secure from outside hackers.

Concurrency and Events : Although all of the current users support multiple users, our current concurrency model is very naive. We plan to build a more robust system modeled on that of Jupiter [14]. We also plan to build a more sophisticated event handler by integrating real world events into the SubArctic user-interface event model.

Real World affordances : Time and thought need to be spent on generalizing the ways in which the affordances of real objects might be extended and providing a toolkit to build these interfaces. The toolkit's development is complicated by the need for physical interaction devices, and a complete toolkit should probably come with sensors and "physical interactors" as well as virtual or "computational" interactors.

A real home : In order to really test the effectiveness of the Domisilica system, we need to integrate it with a real home. We currently have one volunteer who has already implemented a distributed system in his home. We plan to begin integrating with his home this spring.

Domisilica represents a potential solution to the problem of providing ubiquitous access to the home. By modeling the home in a database, we can extend the capabilities of real world objects. This requires us to then integrate an interface to the new services into the residence being modelled. A computer model also makes remote access to services in the

home. In order to provide ubiquitous access to these services, we need to support a broad range of interfaces and users. We have built some toolkits to support the development of multiple interfaces. Now that we have a working system, we look forward to the most exciting phase of this project: putting it into use in real homes. Such a system will allow us to explore activities such as communication and inventory and resource management.

ACKNOWLEDGMENTS

The authors would like to thank colleagues in the GVU Center, particularly those involved in the Future Computing Environments group, and the Broadband Telecommunications Center, for their support and brainstorming that lead to the Domisilica project. Special thanks to Ken Calvert, Chris Atkeson and the many undergraduate students who have contributed to various stages of development of Domisilica and its precursor, CyberFridge. Many thanks also to Joe Bayes, and the many other people who have helped to augment Jen's hands. Thanks also to Ben for help with this submission. This work has been sponsored in part by a grant from Intel Corporation. Jennifer Mankoff is supported by a National Science Foundation HCI Traineeship Fellowship.

REFERENCES

1. Cebus. Available at http://www.cebus.com/.

2. X10. Available at http://www.hometeam.com/x10.shtml. Also see ftp://ftp.scruz.net/users/cichlid/public/x10faq.

3. P. Curtis and D. A. Nichols. Muds grow up: Social virtual reality in the real world. In *Proceedings of IEEE Computer Conference '94*, pages 193–200. IEEE, 1994.

4. W. K. Edwards and E. D. Mynatt. An architecture for transforming graphical interfaces. In *Proceedings of UIST '94*, pages 39–47. ACM, 1994.

5. S. Feiner, B. MacIntyre, and D. Sellgmann. Knowledge-based augmented reality. *Communications of the ACM*, 36(7):53–61, July 1993.

6. G. W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Communications of the ACM*, 36(7):39–49, July 1993.

7. G. W. Fitzmaurice, H. Ishii, and W. Buxton. Bricks: Laying the foundations for graspable user interfaces. In *Proceedings of CHI '95*, pages 442–449. ACM, 1995.

8. S. Hudson and I. Smith. Subarctic user's manual. Available at http://www.cc.gatech.edu/gvu/ui/sub_arctic/sub_arctic/doc/users_manual.html.

9. H. Ishii and B. Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *Proceedings of CHI '97*, pages 234–241. ACM, 1997.

10. T. G. Kieninger. The ”growing up” of hyperbraille - an office workspace for blind people. In *Proceedings of UIST '96*, pages 67–73. ACM, 1996.

11. R. Kraut, W. Scherlis, T. Mukhopadhyay, J. Manning, and S. Kiesler. The homenet field trial of residential internet services. *Communications of the ACM*, 39(12):55–63, December 1996.

12. J. Mankoff and B. Schilit. Supporting knowledge workers beyond the desktop with palplates. In *Proceedings of CHI '97*, pages 550–551. ACM, 1997.

13. M. C. Mozer, R. H. Dodier, M. Anderson, L. Vidmar, R. F. C. III, and D. Miller. The neural network house: An overview. Technical report, University of Colorado, 1996. Available at http://boulder.colorado.edu/lucky/projects/House/house-overview.ps.

14. D. A. Nichols, P. Curtis, M. Dixon, and J. Lamping. High-latency, low-bandwidth windowing in the jupiter collaboration system. In *Proceedings of UIST '95*, pages 111–120. ACM, 1995.

15. D. Norman. *Psychology of Everyday Things*. Basic Books, 1988.

16. T. Raman. Emacspeak – a speech interface. In *Proceedings of CHI '96*, pages 66–71. ACM, 1996.

17. A. Savidis and C. Stephanidis. Developing dual user interfaces for integrating blind and sighted users: the homer uims. In *Proceedings of CHI '95*, pages 106–113. ACM, 1995.

18. A. Venkatesh. Computers and other interactive technologies for the home. *Communications of the ACM*, 39(12):47–54, December 1996.

19. M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.

20. M. Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–83, July 1993.

21. P. Wellner. Interacting with paper on the digitaldesk. *Communications of the ACM*, 36(7):87–96, July 1993.

22. P. Wellner, W. Mackay, and R. Gold. Computer-augmented environments: Back to the real world. *Communications of the ACM*, 36(7):24–26, July 1993.

23. N. Yankelovich, G.-A. Levow, and M. Marx. Designing speechacts: Issues in speech user interfaces. In *Proceedings of CHI '95*, pages 369–376. ACM, 1995.